RANDOM SAMPLING AS A BUILT-IN FUNCTION FOR DATABASE ADMINISTRATION AND REPLICATION

Cross-Reference to Related Application

This application is related to U.S. application serial number unknown, filed together with this application, entitled Partition Boundary Determination Using Random Sampling on Very Large Databases, attorney docket IBM 2 0003.

Background of the Invention

The invention pertains to partition size analysis for very large databases having multiple partitions and, more particularly, to accurate, fast, and scalable characterization and estimation of large populations using a random sampling function that is integrated directly into a database engine.

Databases provide a means to conveniently store and retrieve a wealth of information such as, in the business setting, individual and corporate accounts and, in the business example provide a means to analyze business trends and make other business, educational, and scientific decisions. Accordingly, over the years, typical database populations reach upward of a billion rows and records.

Analysis of these large databases for administration and replication purposes

typically involves processes which are very input/output intensive, as numerous queries must be performed by an analysis program across a vast number of records. Random sampling by an associated application program outside of the database management system (DBMS) can reduce the number of records analyzed. However, the number of requests passed from an analysis program to the DBMS remains high because requests must be made not only for selected records but also to skip non-selected records.

It would be beneficial to provide a method and system for administration and replication of large databases including a means for partition size analysis that reduces the amount of time required to perform the analysis so that such analyses can be executed in a more timely basis without placing an undue burden on the computer system hosting the database. The ability to perform size analyses in a timely basis allows database managers to monitor growth patterns and to accurately estimate needs for database reorganization in terms of predicting the time of a required reorganization and projecting space allocation requirements.

Partition size analyses require only a sufficiently accurate approximate solution as compared to the very precise solution obtainable by analyzing each and every item of data in a database. It is of little worth to provide a precisely accurate solution for a volatile database that is constantly changing including changing at the very moment that it is being analyzed. It is typically not possible to provide an exact analysis without first removing a database from online for an extended period of time. For size analyses, only a small portion of the full set of data must be processed to provide an accurate estimate of partition size, especially for very large homogeneous databases.

The present invention provides a method and system for performing database characterization and approximation analyses to generate very precise, as well as timely results. The method is based on first deriving a random sample of known size from a database of unknown size, or known size, and then extrapolating the results to provide an accurate approximation of a full-scale analysis.

The method and system provided are unique in that a random sample is selected of predetermined known size, but uniformly distributed across the entire database, from a database of known or unknown size while reading only a fraction of the records in the database without the requirement of indexing the entire database which, as indicated above, is time consuming and provides results having an unnecessary degree of precision. The sampling facility is provided as a built-in feature of the database management system and not simply attached to the DBMS as an associated external application. This enables earlier pruning and better performance because the sampling function is closer to the source database.

Other previous random sampling techniques typically require that the database be indexed in order not to read the entire database, or read the entire database and randomly select samples from the entire result. As an example, U.S. Patent Number 5,675,786 provides a simple sampling function in a database engine. The sampling function taught there generates a sequential stream of initial results by first addressing a query to the database and then sampling the stream of initial results to produce a sampled result substantially smaller than the initial result.

The present invention, on the other hand, retrieves only a user-selectable fraction

of the records stored in the database. This advantageously improves the overall performance of the system and accuracy of the results.

In order to produce samples of predetermined size that are normally distributed across a database typically requires a knowledge of the exact number of records in the database beforehand. As an alternative to prior knowledge of the number of records, a complete scan of the database prior to sampling is needed. For example, the '786 patent identified above requires that a particular sampling probability be selected in order to produce a particular sample size from a given result. The present invention, however, overcomes this requirement.

The present invention therefore provides a solution to the aforementioned problems, and offers other advantages over the prior art.

Brief Summary of the Invention

In accordance with the present invention, a system and method for administration and replication of database partitions is provided. The administration system includes a built-in (integrated) sampling facility that performs a random sampling function to sample a predetermined number of records from the database using a preferred random sampling algorithm. Preferably, each time the method is utilized, different random number generator seed values are used so that different database records are selected for the random sampling. Further, the selected records are different for successive utilizations of the method when at least one record has been added to or deleted from the database between successive utilizations of the random sampling method.

Statistics are stored for each of the sampled records, including a record key for each record, and an approximation partition analysis is produced based on the stored statistics. The approximation partition analysis is not mathematically exact because of the sampling, however, the analysis is generally accurate to within a few percentage points which is adequate for dynamically changing databases where exactness would be rendered meaningless after a few transactions.

The preferred embodiment of the sampling method follows a mechanical procedure including the following steps:

1. A table of number pairs (Y_j, I_j), $j=1,2,...,S$, is generated where all Y and all I are initially set to zero, and S is the user-selected predetermined sample size.

2. A reservoir for storing records is set to an empty state.

3. Variable M, an index to the reservoir, is set equal to zero.

4. A sequence of N non-repeating random numbers $U_1, U_2, ..., U_N$, $0<U<1$, is generated as database records are considered for retrieval, where N is the initially unknown number of records in the database, and additional steps are performed for each random number U_k generated including:

4.1 The next record in the database is skipped if U_k is less than the smallest value of Y in the table of number pairs.

4.2 The table is updated if a Y less than U_k exists as follows:

4.2.1 M is set equal to M+1.

4.2.2 The smallest Y in the table is replaced with U_k.

4.2.3 The I value paired with the smallest Y is set equal to M.

4.2.4 All or part of the next record of the database is stored in the reservoir where the current value of M is an index to the stored record.

4.2.5 The table is rearranged into a heap with respect to Y.

Following the generation of an approximation partition analysis, multiple partition boundaries are defined that are sufficient to accommodate the database records and include spare space for future growth. All database records are accessed in an arbitrary sequence and the partitions are filled iteratively, except the last, with the accessed records to a maximum byte count, and the remaining accessed records are stored in the last partition.

Raw partition analysis, without random sampling analysis, places a heavy strain on a computer system in terms of memory usage and typically requires multiple dataspaces. Random sampling relieves the strain on the computer system in terms of processing and memory requirements. Much less memory is required to analyze 20,000 sampled records using the random sampling approach than to analyze 2,000,000,000 records without sampling. However, in order to maintain consistency with an unsampled approach which may be desirable under some circumstances, the preferred method using random sampling analysis utilizes one or more of each of the following types of dataspaces: index, key and statistics.

One benefit obtained from the present invention as a result of providing a built-in sampling facility is the reduction in the number of system calls required to perform an approximation partition analysis.

Another benefit obtained from the present invention is the reduction in time

required to perform an approximation partition analysis compared to the time required for an exact partition analysis.

Still another benefit obtained from the present invention is that approximation partition analyses is performed frequently without straining or otherwise compromising computer system resources.

Yet another benefit obtained from the present invention is an improved accuracy of the analyses, particularly for homogeneous database populations.

Yet another benefit obtained from the present invention is that a random sample of predetermined size is obtained without prior knowledge of the number of records in the sampled database.

Other benefits and advantages of the subject method and system will become apparent to those skilled in the art upon a reading and understanding of this specification.

Brief Description of the Drawings

The invention may take physical form in certain parts and steps and arrangements of parts and steps, the preferred embodiments of which will be described in detail in the specification and illustrated in the accompanying drawings hereof and wherein:

FIGURE 1 is a generalized diagram of a computer system having a partitioned database and a database management system; and,

FIGURE 2 is a flowchart illustrating the preferred method of performing random sampling on very large databases in accordance with the present invention.

Detailed Description of the Invention

The capacity of DL/I databases is limited by the maximum size of a data set that can be addressed by a four-byte relative byte address (RBA). Many other databases in use presently suffer from similar size limitations. In current full function databases managed by database management systems such as IMS, multiple data sets are supported. This helps to increase the capacity of the database. One requirement, however, is that all segments of the same type must be in the same data set. As a result, when one data set is full, the database is deemed to be essentially full even if empty space exists in the remaining data sets. As a consequence, methods have been developed to extend the capacity of such databases.

As shown in Figure 1, partitioning removes the data set limitation by relieving the restriction that all occurrences of the same segment type must be in the same data set. Partitioning database **10** groups database records into sets of partitions **12** that are treated as a single database by database management system (DBMS) **14** running on computer system **16**, while still permitting functions to be performed independently against individual partitions.

Partitions may be very large. More particularly, each partition has the same capacity limit as a non-partitioned database, in that no single data set may exceed the addressing limit. The ability to divide the database into multiple partitions provides the added advantage of decreasing the amount of data unavailable if a partition fails or is taken offline. For these reasons, partitioning has become a widely used and is well known means of extending the storage capacity of databases.

Correct initial sizing of a large partitioned database is important because after the database is in use, reorganization of the database is usually necessary to alter the size of the database, resulting in a potentially extended periods of database unavailability. For this reason, computer utility programs are used to statistically analyze existing databases while they are in use for growth trends and future storage capacity requirements.

A typical partition analysis program **18**, in communication with DBMS **14**, performs queries and stores compressed statistics in random access memory (RAM) **20**, which may comprise real and virtual memory, or an external storage device **22** for every record in the database. Under certain conditions, the analysis program sorts those statistics by key in RAM, or on the storage device. Since a database may contain upward of a billion records, storing and sorting even compressed statistics involves numerically intensive computation, which may burden even large powerful computers such as a mainframe. Sorting a large number of statistical records places a heavy load on the computer system because the time for sorting is typically proportional to $N_{rec}*\log_2(N_{rec})$ where N_{rec} is the number of records being sorted and $\log_2$ is logarithm base **2**. Even when sorting is not needed, the analysis is time consuming because, as appreciated by those skilled in the art, nearly all elapsed time is spent in a get-next function to traverse the entire database.

By way of example, in the case of an IMS database which can contain up to 8 gigabytes (GB) in keys, on a computer system having RAM **20** dataspaces of up to 2^{31} bytes (2GB), four dataspaces are required to store the keys. Another 2GB are sufficient to store indices to the keys. However, the record statistics, even when compressed, may require dozens of

dataspaces. To minimize the effort of storing and sorting, the present invention randomly samples a database and produces an extrapolated partition analysis **24** providing sufficiently accurate results. Preferably, the sample size selected is sufficiently small so that three dataspaces will suffice, one each for indices, keys, and statistics.

An analysis program **16**, in communication with DBMS **14**, partitions a random sample size of S records, and then scales the tabulated numbers by the ratio of the number of records in the database to the number of records in the sample. For a homogeneous database, with records mostly of similar structure, the sampling is remarkably accurate. In one experiment used to test the accuracy of the sampling generally, 20,000 random integers were selected from a set of integers 1 to 2,147,483,646, the sample sorted, and the middle two averaged (sample[10000] and sample[10001]). Repeated 100 times, this experiment produced estimates all within 1.78% of the actual mean, 1,073,741,823.5.

In the present invention, an IMS database is bisected by first sorting by key, and then bisecting at the median. A sample of size 20,000 normally produces a result within a percent or two of the exact result which would be obtained if the entire database was sampled. The present invention leverages the sampling strategy that in most cases it doesn't make sense to store and sort orders of magnitude more data for less than 2% improvement in accuracy, particularly when the database is not static. Perhaps counter-intuitively, larger databases do not require larger samples for similar accuracy. Accordingly, the present invention does not rely on large sample sizes for accuracy but rather is based on the theory of "order statistics" to perform random sampling of records. Although any suitable form of order statistics can be used, the

complete mathematical description of the preferred order statistics used in the present invention is taught in "Introduction to Mathematical Statistics" by Hogg and Craig, 5th ed. the teachings of which are incorporated by reference.

The scaling factor used to inflate the sample to the size of the full database is the number of records in the database (N) divided by the number of records in the sample (S), thus (N/S). Therefore, by definition, the root segment counts in the estimated partitions add up exactly to the total number of root segments in the database, while other estimates merely approximate corresponding database totals. Preferably, sampling stability is verified beforehand by experimenting with different sample sizes and random seeds, e.g. "sample=10000,seed=7".

While random sampling can be of great benefit in reducing the number of records retrieved from database **10** for partition analysis, a large amount of time is spent in communicating requests from analysis program **18** to DBMS **14**. For example, a call to DBMS **14** must be issued for each record to be skipped as well as for each record to be retrieved from database **10**. In accordance with the present invention, a more efficient solution is to provide a built-in random sampling facility **26** configured as a part of the DBMS **14**. In that way, only a single request from analysis program **18** is required to provide parameters to DBMS **14** for random sampling. Sampling facility **26** then performs all random sampling tasks on the database **10**, and stores statistics from sampled records in RAM **20** or on external storage device **20** for use by analysis program **18**.

An added benefit of providing the built-in sampling facility in accordance with the present invention, is that sampling facility **26**, as an integral part of DBMS **14**, has access to

all low level I/O functions and I/O buffers. This enables rapid access to records being retrieved and a more efficient means for skipping records not selected for retrieval.

Sampling a desired sample size requires selecting S items at random from database **10**. However, the number of records present in the database is unknown. One method is to traverse the database with the sampling facility **26** and count the records, then take a second pass to make the random selections. In accordance with the preferred embodiment of the invention, however, M records ($M \geq S$) are sampled on the first pass, where M is much less than N, so that only M items must be considered on the second pass. It is necessary to do this in such a way that the final result is a truly random sample of the original file.

With reference now to Figure 2, and with continuing reference to Figure 1, the preferred algorithm of the built-in random sampling facility **26** is illustrated for providing a method of random sampling according to the aforementioned conditions. The algorithm is incorporated directly into the sampling facility **26** so that statistics for sampled records are communicated by DBMS **14** to analysis program **18** which, in turn, produces an approximation partition analysis report **24**. The approximation partition analysis is then used by reorganization program **28** to reorganize input database **10** by reading records from partitions **12** and writing the same records to an output database **30** comprising partitions **32**, including first partition **34**, intermediate partitions **36** and last partition **38**.

The overall strategy incorporated into sampling facility **26** preferably uses the technique of order statistics. N random values are computed, and then the largest S of these is ascertained. The corresponding S records are selected for the final sample. During the first pass,

a reservoir is constructed which contains only those M records which are possible candidates, i.e., those records which correspond to a random value that has not been preceded by S larger values. The first S items are always placed into the reservoir.

The sampling algorithm first performs several initialization functions. In a first step **40**, a variable S is initially set to a pre-configured default sample size. A particular desired sample size is selectively received to replace the initial value of S at step **42**. A table of paired numbers (Y_j,I_j) is generated and initialized such that $(Y_j,I_j)=(0,0)$, $j=1,2,...,S$ at step **44**. A reservoir **R**, for storing compressed statistics from selected database records, is initialized to an empty state at step **46**. The variable M which represents the number of possible coordinates is zeroed at step **48**, and a random number generator seed value is provided at step **50**.

Once initialization has been completed according to the above, an iterative loop is processed to perform the random sampling function. A random number U is generated from a random number generator capable of generating N uniformly distributed, non-repeating random numbers at step **52**. A set of variables (Y_k,I_k) is found such that $Y_k \leq Y_j$, $1 \leq j \leq N$, or, in other words, Y_k is a minimum Y at step **54**. Next, the random number **U** is compared to Y_k at step **56**. A comparison "if $U<Y_k$" is made, and if true, then the next available record in the database is skipped at step **58**, otherwise the variable I_k is tested at step **59** to determine if the minimum Y found at step **54** is residue from the reservoir initialization process. This indicates that the reservoir is not yet full. If it is determined at step **59** that the reservoir is not yet full, the variable M is incremented M=M+1 at step **60**, and the next available record in the database, or portions of it, are stored in reservoir **R** where M is an index, symbolically R_M at step **62**; the variables Y

and I are updated $(Y_k,I_k)=(U,M)$ at step **64**, and the table of paired numbers (Y,I) is rearranged to form a heap (defined below) with respect to Y at step **66**.

A test is then performed to determine if more records exist in the database at step **68**, and if more records exist, processing returns to the top of the iterative loop, otherwise processing continues to the next step.

After the reservoir is full, the minimum Y value found at step **54** will not be a value remaining from the initialization step **44**, but will be a non-zero value originating from the random number U assigned previously at step **64**. This is tested at step **59** whereupon the value of I_k is assigned to the variable M at step **61**. It can be seen that the variable M increases by integer increments until the reservoir becomes full. Thereafter, the increment step **60** is bypassed whereupon the indexes M are revised or reassigned to the reservoir entries. As a final step, the table of number pairs (Y,I) are sorted on I to place the table in order by record at step **70**.

For purposes of the description of the preferred embodiment of the invention, the aforementioned heap is defined as follows:

S elements (Y,I) are a heap with respect to Y if and only if $Y_j<Y_{2j}$ and $Y_j<Y_{2j+1}$ for all $j\leq S/2$.

Maintaining the table in a heap, so that the smallest element is always on top, facilitates testing each random number against the smallest number in the table. Each time a random tag is found that is larger than the smallest Y in the table, the smallest element is replaced, and the table is reformed into a heap.

The memory required by a partition analysis, even when random sampling is

employed, can be large and, consequently, multiple dataspaces may be required. For databases organized with indexes and keys, sampling may require one or more dataspaces, e.g. one or more index dataspaces, one or more key dataspaces, and one or more statistics dataspaces.

After the random sampling has been performed by sampling facility **26**, and analysis program **18** has performed a partition analysis, reorganization program **28** defines output partitions, accesses all database records in an arbitrary sequence and iteratively fills all of the partitions, except the last, to their maximum byte count. The last partition is typically filled to less than its maximum byte count. The partitions can optionally be sized somewhat larger than the calculated maximum byte count to allow for growth within each partition.

The invention has been described with reference to the preferred embodiments. Modifications and alterations will occur to others upon a reading and understanding of the specification. It is our intention to include all such modifications and alterations insofar as they come within the scope of the appended claims, or the equivalents thereof.